УДК 004.05
DOI: 10.31673/2412-9070.2022.042933

**В. Д. СОЛОДКИЙ-СОЛОДАРЕНКО**, аспірант,
Державний університет телекомунікацій, Київ

# ПОРІВНЯЛЬНИЙ АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ КОМП'ЮТЕРНИХ СИСТЕМ

***Хмарні застосунки на вимогу є однією з технологій, що стрімко розвиваються та користуються великим попитом на ринку. Зі збільшенням кількості комп'ютерних систем і вимог до бізнес-аналітики в реальному часі потенційні клієнти таких послуг звертаються до постачальників послуг «обчислення на вимогу». У ситуації, коли клієнт надсилає запит у роботу, цей запит розгортається на комп'ютерній системі в центрі оброблення даних постачальника послуг. При цьому немає стовідсоткової впевненості, що завдання буде виконано, а не провалено через брак ресурсів або через відмову програмного забезпечення. Проблеми відмовостійкості та високої доступності зараз є найбільш гострими в цій сфері. Проаналізовано та зіставлено сучасні технології відмовостійкості та високої доступності. Порівняно сучасні моделі системи високої доступності та інструменти для її впровадження. Виявлено недоліки сучасних моделей та засобів упровадження відмовостійкості. Розроблено рекомендації щодо застосування підходів, які допоможуть усунути розглянуті недоліки в різних випадках.***

**Ключові слова:** відмовостійкість; висока доступність; балансування навантаження; хмарні обчислення; хмарний застосунок; інфраструктура як послуга; застосунок як послуга; комп'ютерна система; бізнес-аналітика; центр оброблення даних; контейнеризація; рівень обслуговування; масштабування.

## *ВСТУП*

«Обчислення на вимогу» є однією з послуг, що стрімко розвиваються та мають широкий попит на сучасному ринку інформаційних технологій. Зі збільшенням кількості комп'ютерних систем і вимог до бізнес-аналітики в реальному часі потенційні клієнти таких послуг звертаються до постачальників сервісів «обчислень на вимогу» [1]. У центрах оброблення даних постачальників таких послуг комп'ютерні системи стикаються з великим обсягом оброблюваних даних, що, зі свого боку, призводить до дисбалансу в навантаженні та провокує відмови в системах. Відмовостійкість як форма високої доступності дає змогу кінцевим користувачам отримувати доступ до комп'ютерної системи в разі відмови одного чи кількох вузлів. Водночас висока доступність — це характеристика, яка описує підтримання доступності всіх систем за допомогою автоматизованих механізмів відновлення, після події відмови вузла системи, передавання інформації та закладених у систему функцій [2].

Сьогодні існує велика кількість комп'ютерних систем, які не відповідають сучасним стандартам рівня обслуговування. Саме *Service Level Agreement* (SLA — угода про рівень обслуговування) і спрямований на визначення рівня обслуговування, на який очікує клієнт від постачальника послуг, установлюючи показники, за якими ця послуга вимірюється [3]. Більшість наявних моделей оптимізації балансують навантаження через оптимізацію алгоритму вибору оптимального хоста, але така модель є неефективною в довгостроковій перспективі. Тож лише впровадження відмовостійкості в систему може допомогти в розв'язанні цієї проблеми.

## *ОСНОВНА ЧАСТИНА*

**Постановка задачі.** У процесі використання інфраструктури як послуги (IaaS) клієнти очікують на ефективне виконання своїх задач за оптимальну ціну. Коли клієнт надсилає задачу в роботу, вона розгортається на комп'ютерній в центрі оброблення даних постачальника послуг. Зазвичай оброблення задачі відбувається на фізичному вузлі випадковим чином [4]. Завдяки такому підходу є змога виокремити вузол із недостатнім обсягом ресурсів, що може призвести до затримки в обробленні даних та частих збоїв.

Відмовостійкість завдяки програмним інструментам допомагає підвищити рівень доступності системи до затверджених у SLA значень і дає можливість упровадити систему в стан неперервної експлуатації. Переважно стратегія неперервності бізнесу охоплює як високу доступність, так і відмовостійкість, щоб гарантувати підтримання організацією основних функцій системи як і під час незначних проблем, так і в разі критичних збоїв системи. Важливими критеріями оцінювання доступності системи в процесі проєктування та планування відмовостійких систем високої доступності є такі характеристики:

• ***час простою*** — відмовостійка система має мати мінімально допустимий рівень перерв у обслуговуванні. Наприклад, система з доступністю 99,999% не працює приблизно на 5 хв на рік, при цьому очікується, що відмовостійка система працюватиме неперервно без неприйнятних перерв у обслуговуванні;

• ***сфера застосування*** — відмовостійкість ґрунтується на наборі ресурсів та компонентів, які спільно використовуються для керування збоями та мінімізації часу простою. Наприклад, відмовостійкість залежить від резервного джерела живлення, а також апаратного та програмного забезпечення, яке може виявляти збої та миттєво переходити на резервні вузли;

• ***вартість*** — відмовостійка система може бути дорогою, оскільки потребує організації неперервної роботи, яка охоплює обслуговування додаткових резервних компонентів та ресурсів. Відмовостійкість зазвичай постачається як частина загального пакета від вибраного постачальника сучасних послуг та технологій відмовостійкості.

Більшість систем потребують розроблення індивідуальної структури відмовостійкості, тоді як для простіших систем може бути достатнім використання наявних технологій забезпечення відмовостійкості [5]. У процесі проєктування системи слід зважати на стійкість кожної системи до відмов у наданні послуг, вартість таких відмов, SLA між постачальником технології відмовостійкості та кінцевим користувачем, а також на вартість і складність упровадження відмовостійкості.

***Метою роботи*** є порівняльний аналіз технологій підвищення відмовостійкості комп'ютерних систем, визначення недоліків, розроблення рекомендацій щодо застосування підходів, які уможливлять їх усунення.

### *Огляд сучасних моделей забезпечення відмовостійкості*

Висока доступність системи зазвичай вимірюється як відсоток безвідмовної роботи протягом року. Показник у 100% використовується для позначення сервісного середовища, в якому немає простоїв або збоїв взагалі. Задля досягнення того чи іншого рівня високої доступності та відмовостійкості комп'ютерної системи послуговуються моделями, які різняться між собою топологіями, поведінкою під час події відмови основного вузла системи, організацією спільного сховища інформації, часом переходу після події відмови основного вузла тощо [6]. Та чи інша модель високої доступності застосовується в процесі побудови відмовостійкої системи, зважаючи на її потреби та порядок застосування.

Топології також класифікуються за рівнем забезпечення відмовостійкості [7]. Нині можна виокремити п'ять топологій, які використовуються у високодоступних системах:

**1. N+N** — під час застосування такої топології високої доступності в комп'ютерній системи є два основних (дубльованих) вузла і немає резервного. У разі події відмови одного з вузлів інший вузол продовжує свою роботу, незважаючи на втрату зв'язку із сусіднім вузлом. Після повернення втраченого вузла потрібно провести процедуру синхронізації даних із працюючим вузлом для продовження коректної роботи обох вузлів. Така топологія є найпростішою в підтриманні та імплементації, але не може забезпечити потрібного рівня відмовостійкості.

**2. N+1** — вторинний вузол (резервний вузол), активується, щоб перебрати на себе роль несправного вузла (основного вузла). Резервний вузол має бути сконфігурований так, щоб мати змогу виконувати функцію будь-якого з основних вузлів. Таку топологію рекомендовано використовувати для систем, які одночасно здійснюють небагато задач. Ця топологія відрізняється простотою та дешевизною, але водночас не може забезпечити належного рівня високої доступності для великих систем.

**3. N+M** — якщо систему використовують для реалізації багатьох задач, резервний вузол не завжди може забезпечити виконання всіх функцій, що зазнали відмови в роботі. У такому разі потрібно сконфігурувати більше одного вторинного вузла, які матимуть змогу розподілити між собою функції основних вузлів у разі їх відмови. Таку топологію рекомендовано для застосування в системах, які одночасно виконують велику кількість різних задач і можуть забезпечити відмовостійкість під час значних навантажень, але з надійністю топології зростає й ціна її імплементації та підтримання.

**4. N-до-1** — вторинний вузол (резервний вузол) стає активним, тобто тимчасовою заміною одного з основних вузлів доти, доки його не буде відновлено. Після відновлення основного вузла системи він може зайняти місце резервного вузла для того, щоб не витрачати час на передавання ролі, а одразу відновити відмовостійкість системи. Топологія за своєю організацією схожа на N+1, але в ній резервний вузол перебирає на себе всі функції основного вузла.

**5. N-до-N** — під час використання N-до-N топології функції несправного вузла розподіляються між іншими основними вузлами. Потреба в резервному вузлі усувається, але з'являється необхідність у резервації додаткових потужностей у кожному активному вузлі. Топологія N-до-N є комбінацією топологій N+N і N+M та є компромісною з погляду витрат на імплементацію та підтримання.

Імплементація високодоступної системи може відбуватися різними способами, і традиційні моделі відмовостійкості описують поведінку системи в разі події відмови основного вузла системи [8].

Характеристики базових моделей імплементації відмовостійкості наведено в табл. 1.

*Таблиця 1*

**Базові моделі імплементації відмовостійкості**

| Модель | Топологія | Поведінка резервних вузлів системи | Організація сховища даних | Час переходу після події відмови основних вузлів |
|---|---|---|---|---|
| Балансування навантаження/ Load balancing | N+N, N-до-N | Усі вузли активні та обробляють запити одночасно | Дані репліковані та синхронізовані на всіх вузлах системи | 0 с |
| «Гаряче» очікування/ «Hot» standby | N+1, N+M, N-до-1 | Програмне забезпечення встановлено і на основних, і на резервних вузлах системи, резервний вузол увімкнено, але не обробляє запити, допоки основний вузол не вийде з ладу | Дані репліковані та синхронізовані на всіх вузлах системи, але їх оброблення здійснюється лише на основних вузлах | 30-60 с |
| «Тепле» очікування/ «Warm» standby | N+1, N+M, N-до-1 | Програмне забезпечення встановлено і на основних, і на резервних вузлах системи, резервний вузол вимкнений та не обробляє запити. У разі відмови основного вузла резервні вузли автоматично починають свою роботу як основні вузли відповідно до заданих раніше інструкцій | Дані зберігаються на зовнішньому спільному сховищі | 10-15 хв |
| «Холодне» очікування/ «Cold» standby | N+1, N+M, N-до-1 | Резервні вузли є ідентичними копіями основних вузлів. Резервний вузол встановлюється лише в разі першої події відмови основного вузла. Під час подальших відмов основних вузлів системи резервні вузли починають свою роботу як основні вузли за участі оператора системи | Дані зберігаються лише на основному вузлі системи. Дані основного вузла системи відновлюються на резервному вузлі за участі оператора системи | 2-4 год |

Варто зазначити, що час відновлення роботи системи може також залежати від конфігурації комп'ютерної системи, її елементів та загалом задіяного програмного забезпечення. Деякі характеристики системи не можуть бути чітко стандартизовані і спрогнозовані [9].

***Порівняльний аналіз програмних інструментів забезпечення відмовостійкості систем***

Сучасний спектр програмних інструментів забезпечення відмовостійкості є доволі широким. Поверхнево, усі вони покликані забезпечити безвідмовну роботу комп'ютерної системи, але фактично мають значні відмінності в концептуальному плані [10]. Порівняльні характеристики найбільш поширених інструментів забезпечення відмовостійкості унаочнює табл. 2.

*Таблиця 2*

**Характеристики інструментів забезпечення відмовостійкості**

| Назва | Вартість | Платформа | Можливість балансування навантаження | Кількість резервних вузлів | Можливість «гарячого» масштабування | Можливість використання у хмарі |
|---|---|---|---|---|---|---|
| pacemaker/ corosync | Безкоштовно | Debian-based/ RHEL Linux | Так | Від трьох до 16-ти | Ні | Лімітовано |
| Proxmox VE HA Cluster | Платна і безплатна версії | Proxmox | Ні | До 16-ти | Ні | Не передбачено |
| vSphere HA | Лише платна версія | VMWare | Так | До 64-ти | Ні | Не передбачено |
| Docker Swarm | Безкоштовно | Мультиплат-форма | Так | Рекомендовано до семи | Так | Лімітовано |
| Kubernetes | Безкоштовно | Мультиплат-форма | Так | Рекомендовано до 5000 | Так | Так |

***Висновки***

Отже, у результаті аналізу можна дійти висновку, що більша частина поширених сьогодні інструментів забезпечення відмовостійкості обмежені у своєму функціоналі. Так, наприклад, кластерна зв'язка pacemaker/corosync хоч і має потрібні функції балансування навантаження та забезпечення відмовостійкості, проте не може бути задіяна у великих кластерах, має обмеження у виборі операційних систем та не є рекомендованою до застосування в хмарах. Комерційні рішення, такі як Proxmox VE HA та vSphere HA, використовують власні операційні системи, розроблені для невеликих кластерів, мають вузьку сферу застосування і не можуть бути використані як рішення для SAAS-застосунку. Водночас сучасні мультиплатформні технології Docker Swarm і Kubernetes розроблені саме для роботи в хмарі з можливістю швидкого масштабування кластера та є повністю відкритим програмним забезпеченням. Цей факт дає змогу розробникам та адміністраторам упроваджувати необхідні зміни під свої застосунки без додаткових ліцензій. Такий підхід дає можливість адаптувати технологію під конкретний хмарний застосунок та впроваджувати модифікації, які підвищують відмовостійкість конкретної комп'ютерної системи. Варто також зазначити, що Docker swarm, на відміну від Kubernetes, не може бути застосовний у великих системах та не має переваг у масштабуванні та керуванні під час використання його у хмарі.

Великому бізнесу доцільно застосовувати комерційні рішення Proxmox VE або vSphere HA для побудови власної хмарної інфраструктури спільно з Kubernetes як оркестратора для застосунку. Такий підхід значно підвищить керованість комп'ютерної системи та забезпечить надійний рівень захисту даних. Середньому і малому бізнесу більш доцільно використовувати послуги хмарних провайдерів, таких як AWS або Azure, де можна розмістити відмовостійкий застосунок за допомогою Kubernetes без побудови власної інфраструктури. Це зменшить керованість комп'ютерної системи, яка розміщує застосунок, але водночас оптимізує витрати та час імплементації.

За результатами проведеного аналізу можна стверджувати, що в майбутньому більшість комп'ютерних систем все ж таки перейдуть у хмарне середовище, де питання відмовостійкості наразі стоїть найбільш гостро. За допомогою відкритих технологій Kubernetes та Docker Swarm можна максимально гнучко впровадити високодоступність у хмарне середовище із застосуванням модифікацій, які істотно підвищать відмовостійкість системи. Нині стоїть питання щодо розроблення платформ, які відповідатимуть вимогам стосовно підтримання технологій автоматизації, імплементації штучного інтелекту, мінімізації затримки під час передавання та оброблення даних та захисту інформації.

***Список використаної літератури***

*1. **A dynamical** and load-balanced flow scheduling approach for big data centers in clouds / F. Tang, L. T. Yang, C. Tang [et al.] // IEEE Transactions on Cloud Computing. 2016. 6(4). P. 915–928.*

*2. **Non-cooperative** power and latency aware load balancing in distributed data centers / R. Tripathi, S. Vignesh, V. Tamarapalli [et al.] // Journal of Parallel and Distributed Computing. 2017. 107. P. 76–86.*

*3. **Hiles A.** The Complete Guide to IT Service Level Agreements: Aligning IT Service to Business Needs (3rd Edition) (Service Level Management). Rothstein Publishing. P. 31–33.*

*4. **Blokdyk G.** High availability: Second Edition - CreateSpace Independent Publishing Platform. P. 62–66.*

*5. **Aashish Khaira.** A book on decision making for event-driven maintenance: Steps Towards High Availability & Low Downtime // LAP LAMBERT Academic Publishing. P. 14–17.*

*6. **Wittig A., Wittig M.** Amazon Web Services in Action // Manning. P. 442–450.*

*7. **Gustavo Rau de Almeida Callou.** Planning of Sustainable Data Centers with High Availability: An Integrated Modeling Approach to Evaluate and Optimize Sustainability, Dependability and Cost of Data Center Systems // LAP LAMBERT Academic Publishing. P. 111–112.*

*8. **Rastogi G., Sushil R.** Analytical literature survey on existing load balancing schemes in cloud computing // IEEE. P. 5–7.*

*9. **Piedad F., Hawkins M.** High Availability: Design, Techniques, and Processes – Pearson. P. 42–47.*

*10. **Hideto Saito, Hui-Chuan Chloe Lee, Ke-Jou Carol Hsu.** Kubernetes Cookbook: Practical solutions to container orchestration, 2nd Edition //Packt Publishing. P. 265–266.*

*11. **Бондарчук А. П.** Розрахунок максимальних значень інтенсивності потоків даних між окремими вузлами інфокомунікаційної мережі // Сучасний захист інформації. 2015. №. 2. С. 89–92.*

*V. D. Solodkyi-Solodarenko*

**COMPARATIVE ANALYSIS OF MODERN FAULT TOLERANCE TECHNOLOGIES**

*On-demand cloud applications is one of the rapidly developing technologies that highly demanded on the market. With the increase of computer systems and requirements for real-time business intelligence, potential customers of such services turn to providers of «computing on demand» services. In the data centers that used for such services, computer systems face a large amount of processed data, which in turn leads to an imbalance in the load and provokes system failures. With infrastructure as a service (IaaS) usage, customers expect efficient performance of their tasks at an optimal price. In situation, when a client submits a task for work, it is deployed on a computer system in the data center of the service provider and there is no a hundred percent assurance that task would be completed and not failed because of lack of resources or software failure. Fault tolerance and high availability issues are currently the most acute in that field. The modern set of software tools for providing fault tolerance is quite wide. In general, they are all called to ensure the trouble-free operation of the computer system, but in fact they have significant differences in conceptual terms. High availability solutions provide fully automated failover to a backup system so that users and applications can continue working without disruption. HA solutions must have the ability to provide an immediate recovery point. At the same time, they must provide a recovery time capability that is significantly better than the recovery time that you experience in a non-HA solution topology. Analyzed and compared modern fault tolerance and high availability technologies. Compared modern models of High Availability system and tools for its implementation. Identified the shortcomings of modern models and tools for fault tolerance implementing. Developed recommendations for the usage of approaches that would help to eliminate reviewed shortcomings in different cases.*

**Keywords:** fault tolerance; high availability; load balancing; cloud computing; cloud application; infrastructure as a service; software as a service; computer system; business inelegance; data center; container orchestration; service level agreement; scaling; data sharing; system deploying; cluster; node.